EXHIBIT 9A

Cisco NX-OS Software: Business-Critical Cross-Platform Data Center OS

What You Will Learn

Modern data centers power businesses through a new generation of applications and services. Virtualization, cloud computing, high-performance computing, data warehousing, and disaster recovery strategies, among others prevalent in the current environment, are prompting a whole new set of requirements for network infrastructure. To meet the needs of the modern data center, a network device - or more particularly, the operating system that powers that device - must be:

- Resilient: To provide critical business-class availability
- Modular: To be capable of extension to evolve with business needs and provide extended lifecycles
- Portable: For consistency across platforms
- Secure: To protect and preserve data and operations
- Flexible: To integrate and enable new technologies
- Scalable: To accommodate and grow with the business and its requirements
- Easy to use: To reduce the amount of learning required, simplify deployment, and ease manageability

Cisco® NX-OS Software is designed to meet all these criteria across all Cisco platforms that run it. This document describes the features of Cisco NX-OS operating system to help you understand how it can meet the needs of your organization.

Building on a Proven Foundation

Cisco NX-OS is a highly-evolved modular operating system that builds on more than 15 years of innovation and experience in high-performance switching and routing. Cisco NX-OS finds its roots in the Cisco SAN-OS operating system used worldwide in business-critical loss-intolerant SAN networks. As a direct result of having been deployed and evolving from nearly a decade in the extremely critical storage area networking space, NX-OS can deliver the performance, reliability, and lifecycle expected in the data center.

Cisco NX-OS is built on a Linux kernel. By using a version 2.6 based Linux kernel as its foundation, Cisco NX-OS gains the following benefits:

- Established and widely field-proven core kernel code
- Efficient multithreaded preemptive multitasking capabilities
- Native multiprocessor and multicore support

The choice of Linux over other operating systems and of the Linux 2.6 kernel over other versions of Linux was strategic. The following are some of the reasons for this choice:

- By inheritance, this implies that NX-OS shares the largest installed base of any UNIX-like operating system in existence. The community development model and widespread use of Linux provide the benefits of rigorous code review, rapid community-based defect resolution, and exceptional real-world field testing.
- The kernel is a near-real-time OS kernel, which is actually preferred over a true-real-time OS for applications such as networking, which may have many parallel critical activities running on a platform.

At its core, Cisco NX-OS is designed to take advantage of distributed platforms to reduce any effects on the data plane during control-plane operations, including software upgrades. Again, on platforms designed to be highly distributed, it uses this same high-availability infrastructure and distributed architecture to deliver fully non disruptive ISSU. The control planes of a distributed system are upgraded without affecting data-plane forwarding, and after the control planes are successfully upgraded, the control-plane portions of any forwarding hardware that can be nondisruptively upgraded are serviced. This approach effectively transforms planned maintenance windows so that they longer automatically imply a service outage. This increased level of continuous operation successfully accommodates business-critical environments, in which little downtime or degradation of service is essential.

Enhanced Usability and Familiar Operation

Cisco IOS® Software is already the recognized leader in internetworking device operating systems. For decades, Cisco IOS Software has been the foundation for routing and switching configuration in all environments. The Cisco IOS CLI has essentially become the standard for configuration in the networking industry.

To reduce the amount of time needed to learn Cisco NX-OS and to accelerate adoption, Cisco NX-OS maintains the familiarity of the Cisco IOS CLI. Users comfortable with the Cisco IOS CLI will find themselves equally comfortable with Cisco NX-OS. In addition, Cisco NX-OS has integrated numerous user interface enhancements on top of the familiar Cisco IOS CLI to make configuration and maintenance more efficient. These are just some of the simple but effective UI enhancements found in Cisco NX-OS:

- Nonhierarchical CLI: Almost any command can be run from any mode. You can run show commands from the interface and global configuration modes. Global commands can be run from the interface configuration mode. The system is intelligent enough to determine the nature of a command and process it regardless of whether the current mode the configuration or execution mode.
- Configuration mode contextual CLI history: Separate CLI command histories are maintained for each configuration mode. Reentry of commands in a given configuration mode is simplified; you can use the up and down arrow to cycle through the command history stored for that particular configuration mode.
- Advanced multilevel and multicommand output piping: The capability to stream CLI output through advanced filters and parsing commands enables complex formatting and manipulation of information for easier parsing and processing.
- More verbose and descriptive status output: The show command output tends to be more informative and less obscure or opaque in Cisco NX-OS, allowing more effective troubleshooting and status monitoring.

Cisco IOS Software users will quickly find themselves familiar with the Cisco NX-OS CLI and its enhancements. Typically, most networking professionals will also quickly find themselves seeking those functional enhancements in other operating systems.

Virtual Device Contexts

Cisco NX-OS also provides the capability to virtualize the platform on which it is running. Using Cisco NX-OS virtual device contexts (VDCs), a single physical device can be virtualized into many logical devices, each operating independently, effectively approximating separate physical devices (Figure 3).